SmartLighting - A platform for intelligent building management

Helder Moreira[1], Gonçalo Correia[1], Manuel Silva[1], André Marques[1], João Barraca[1], Luis Alves[1], Pedro Fonseca[1], Nuno Lourenço[2]

1 Instituto de Telecomunicações, Universidade de Aveiro
P-3810-193 AVEIRO - PORTUGAL
2 Think Control
{ helderm14@ua.pt, goncalodaniel@ua.pt, a34021@ua.pt, marques.andre@ua.pt, jpbarraca@ua.pt, nero@ua.pt, pf@ua.pt, nuno.lourenco@thinkcontrol.pt }

Abstract. This work proposes a solution to endow buildings with efficiency and intelligence, exploiting the advantages of Complex Event Processing techniques and Internet of Things (IoT) principles. This combination allows efficient management of the entire infrastructure, an in particular enabling lighting to be tailored to the users needs. We validate this solution through a prototype implementation, based on wireless sensors and actuator networks that interact with the environment, using standard lightweight protocols designed for IoT. The prototype is based on high performance and real time platforms, and complex methods for analysis of large streams of data. The implementation is applied to a real world scenario, and will be used as the standard solution for management and automation of an existing building.

Keywords: Internet of Things, Complex Event Processing, Wireless Sensor Networks, Building Automation

1 Introduction

A large share of the global energy usage is taken by buildings, whose number and size keeps growing. This creates a need for energy efficient solutions which led to the development of Building Automation Systems (BAS), whose primary goal is to achieve significant energy savings. This is done by efficiently automating several systems inherent to a building such as the lighting and Heating, Ventilation and Air Conditioning (HVAC), CCTV and lighting to name a few. Current solutions focus in full integration, where the automation rules cover multiple subsystems, providing an unified management solution.

In order to allow the integration of the different systems and interoperability between different devices, several standards have emerged, such as BACnet, LonWorks and KNX, with ModBus and OPC, also DALI is the most used standard for lighting control [1].

With the Internet of Things (IoT) revolution, new solutions have been presented, specially designed for constrained devices. IEEE 802.15.4 was one of the first

standards targeting Low Rate WPANs (LR-WPANs), but it only specifies the physical and MAC layers. Thus, 6LoWPAN was standardized to carry IPv6 packets within small link layer frames [2]. Taking advantage of these two standards, other communication technologies have emerged, which is the case of ZigBee and Bluetooth Low Energy (BLE), and in a near future the Wi-Fi HaLow [3].

Regarding higher layer protocols, both MQTT and CoAP are the most commonly adopted for constrained devices, since they offer an extraordinary performance and various features, while working at minimum bandwidth. Moreover, despite not being designed targeting the IoT, AMQP and XMPP have been evolving towards it, and now play an important role on it.

Furthermore, the number of devices used in IoT solutions, namely in smart buildings, are increasing rapidly. This increase creates many issues related to device management, which must be solved considering a vision of autonomous behaviour. For this reason, several standards were created for dealing with device management in IoT systems. The most known device management solutions include TR-069, a technical specification that defines an application layer protocol for remote management of end-user devices and it was published by the Broadband Forum and entitled CPE WAN Management Protocol; OMA Device Management (OMA-DM), a device management protocol specified by Open Mobile Alliance Device Management Working Group. The OMA-DM specifications define the protocols and the mechanisms allowing a server to deliver configuration parameters to a client, by using a defined set of Device Management Commands for various management procedures to be executed inside a well-defined and secure environment (DM Session); Lightweight M2M, a standard that defines the application layer communications between an LWM2M Client (located in a device or gateway) and an LWM2M Server.

The problem with the first two solutions is that they were created to work in Telco environments, and not considering constrained devices, therefore the need for other more lightweight solutions. Moreover, having a device management standard doesn't magically enables device operation in other words, a standard only helps us to define the interactions between devices and service applications. To be able to properly operate millions of devices, a proper object representation is required. This will map devices into objects, enabling rich interactions with many, heterogeneous devices. In this area, the most accepted are the ones specified by ETSI M2M, which tend to be complex, and the ones specified by IPSO, that proposes the LWM2M object specification and is tailored to constrained devices.

Regarding to the automation logic that operates over the sensor objects, this requires mechanisms for analyzing and processing heterogeneous sources, issuing command with very low latency. Examples of low latency processing systems are open-source Apache projects, such as Apache Storm and Apache Spark for, respectively, stream processing and batch processing. An alternative approach, more suited to this environment, considers solutions from the area of Complex Event Processing (CEP), which is the process of analyzing large streams of information from multiple sources and, by detecting patterns and identifying meaningful complex events, quickly infer a conclusion from them and possibly generate an action. It can be very useful in a large variety of applications, by allowing to predict situations and

thus avoiding issues or seizing opportunities [4]. Representative CEP systems are Esper, Drools Fusion and Siddhi, which has evolved to the WSO2 CEP.

Taking in consideration all these technologies and the context of building automation, this work presents a solution for an effective, low latency automation solution, considering most software aspects, as well as sensors and actuators. Due to context of the pilot considered, and because of the low latency challenges presented, there is a focus in lighting systems.

This work is organized in 6 sections, being the first section the current Introduction, the remaining sections go as follows. Section 2 presents the system overview architecture, giving a description of each of the composing elements. Section 3 explains the design principles and implementation details about the presented work. Section 4 describes a test scenario, the results obtained and an analysis of those results. Section 5 presents future work that can be made to improve our work. Finally, section 6 supplies the final conclusions about the developed work.

2 System Conceptual Overview

Solutions for effective automation must consider a multitude of components, at different layers and providing different functionality. The aggregate of these components composes the management platform, which operates in a coordinated manner. A particular difficulty of such systems is that different knowledge areas should be present, ranging from electronics, to communications, data processing and visualization. Standards provide a great advantage as they allow companies to focus in specific areas, as long as the solutions developed use the communication protocols and operational primitives. As the focus of our work was the the development of innovative solutions at multiple layers, we consider the use of standard Machine to Machine protocols and solutions, but developed a conceptual solution and encompasses multiple layers. The resulting conceptual system architecture is depicted in Figure 1, and it considers the following layers: Field, Network, Aggregation and Automation, and Services and Applications.

The Field layer considers the devices located in the building and through which most users interact. These devices can be categorized as sensors, actuators or as hybrid devices that do both sensing and actuation. From a conceptual point of view, this will include any HVAC equipment, electric door locks, lighting fixtures, switches, as well as other environmental and presence sensors, current sensors or even other devices used for indoor location. It should be considered that devices are composed by some processing component (usually a microprocessor), a communication interface and then the actual sensor or actuator. This process power can also be taken advantage to endow each device with some intelligence, enabling autonomous decision making as failsafe mechanisms.

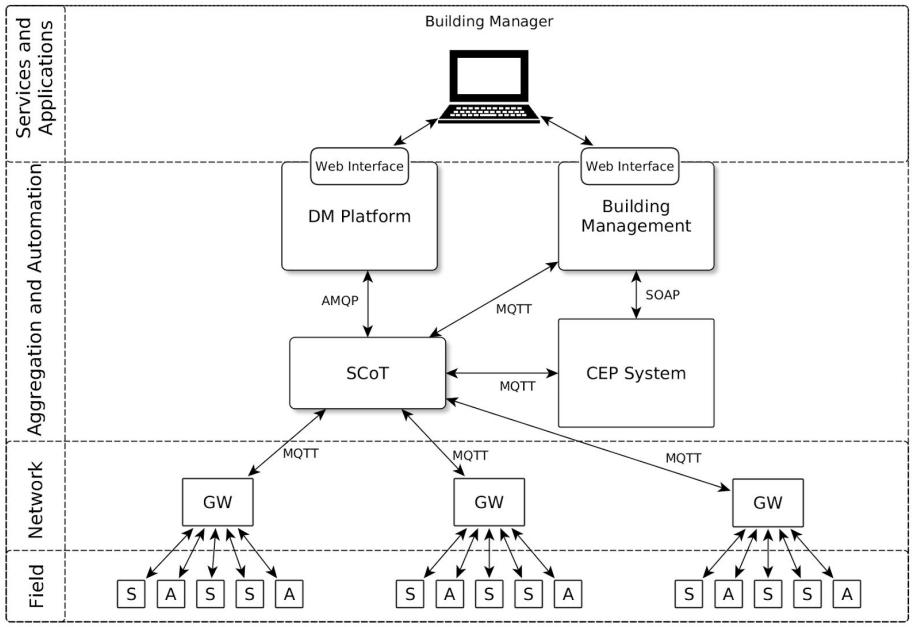


Fig. 1. Architecture diagram.

This topology can be extrapolated to others actuators that may not require such a sophisticated control over its functionalities but nevertheless can have a significant impact in the energy efficiency, and operation of a building. For example, having control over windows blinders can compensate a space lighting with natural light, managing air condition units and windows can translate in lower heating costs.

The Network layer considers the communication interfaces of each Field device, and aggregator nodes named as gateways. These aggregator nodes are more powerful than the Field devices, and will have the capability of interfacing devices with higher layer platform components, in particular the IoT platform. Moreover, they can closely monitor Field devices, discovering new devices or detecting failure. In both cases, they can issue notifications for the higher layers. An important aspect of these components is that they adapt the object oriented automation orders into specific commands tailored to the communication technology. In particular, communication to lower layers can use non-IP protocols, or domain specific protocols, which may be more suited to a particular scenario. Moreover, gateways can host and execute automation rules, as delegates from the automation layer. The interest is to keep automation rules in the absence of higher layers or for emergency and failsafe scenarios. Communication is done through standard M2M/IoT protocols and technologies, such as MQTT, CoAP, Wifi, BLE, ZWave and ZigBee as deemed appropriate.

The Aggregation and Automation layer hosts the components that are capable of receiving IoT data from sensors and actuators, providing mechanisms for storing, distributing and adapting data as needed. In our vision, in particular because we consider a common object representation model, all information at this layer is homogeneous and follows a common structure, even if the specific concent of each data unit can be widely different. Still, all data units are represented by a timestamp and an identifier, which can be used both for purposes of auditing past behavior as for identifying the context of each data.

At its core we consider the existence of the Smart Cloud of Things IoT platform that implements a Machine-To-Machine (M2M) solution, with the goal to connect devices to the cloud. The data sent by the gateways can be retrieved and analyzed by third parties, for example, services, applications, in order to create dashboards with that information [11]. The main purpose of this platform is to provide a standardized, broker based, communication channel, with persistence and standardized APIs, which could be provided by many other solutions.

One of the main components is the Device Management, which tracks all objects (devices) connected to the platform, its status and its properties. In our approach, we have a standardized vision over all connected objects, with a strict structure based on the work from IPSO. Therefore, it is possible to enumerate objects (e.g., a luminary), and interact with it (e.g., turning its light on), in a programmatic manner.

Another main component, and a core aspect of this work, is the CEP engine. This platform is responsible for processing every event generated by sensors (e.g., a switched was activated), and using Complex Event Processing infer actions in real time. Here, an action is not necessarily an action on the environment using actuators. It can also be an alert that can be pushed to another component or directly sent to the building manager. Also, the action can actually be no action as it can be the case of turning on a air conditioning in a room already very cold.

Finally, the building management component is responsible for providing an user-friendly interface for the building administrator to create, edit or delete rules dynamically. Based on these rules it generates the complex code that forms a rule to be applied on the CEP system. Additionally, it is also in this component that the user creates the virtual representation of the building, and distributes the devices by areas in rooms. Thus, this component is also responsible for providing the information about a device's location, i.e to what building, floor, room and area it belongs, and configure the devices accordingly.

The Application and Services layer is the final layer in our architecture, providing programmatic interfaces to other higher layers services, dashboards for users to interact with the building, and analytical tools for alarmistic, management and forensic analysis.

3 Design and Implementation Considerations

Taking in consideration our conceptual architecture, we created a real world implementation, comprising the case of a building with smart lighting. This scenario

was chosen due to the focus of our research groups, automation requirements from actual building owners, allied to the fact that lighting presents a near perfect testcase for validating real time, low latency, heterogeneous systems. Moreover, it can provide real energy savings benefits for building owners, which we also wished to explore. In particular, if we consider a scenario with full softwarization of all existing devices, including the traditional light switches and PIR motion detectors, in order to keep high levels of acceptance, it is vital to deploy solutions able to react with very low latency. Even in the condition where rules are complex and a multitude of parameters is evaluated before a luminary is activated after the user presses the switch. Also, we consider advanced lighting features such as controllable dimming and fade in/fade out.

3.1 Lighting management and actuators

Combining the LED luminaires with a microcontroller serves many purposes, communication and control are the most fundamental but other features may prove useful in integration with the system.

Dimming can be implemented by the microcontroller and it has significant impact in energy saving, LEDs allows multiple dimming techniques but the most popular and efficient ones are definitely analog dimming and pulse width modulation (PWM) dimming and both have advantages and disadvantages that need to be considered when designing an application. Ultimately, choosing one type of technique requires an assessment of the application demands without compromising user comfort.

The microcontroller can be also use to implement logarithmic dimming, accordingly to the Weber–Fechner law, the human eye perceives a linear transition in brightness levels when the change follows a logarithmic logic due to the fact that the eye is more sensitive to changes in lower brightness levels, so the user perceives a linear transition in light levels when the dimming curve is actually logarithmic instead of linear.

Other factors can be customizable, for example a maximum dimming level can be established and adjusted over time to compensate the decay of performance in LED over time.

3.2 Sensor systems

Designing a sensor system requires a full recognition of the building. Division and corridors are individually analyzed and customized according to their needs and characteristics.

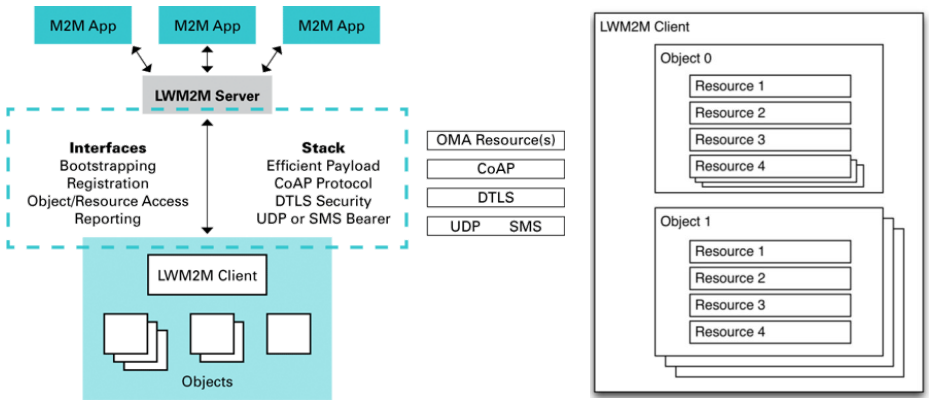
Each luminaire or a small group has a presence sensor, which in case of sensing the presence of a person will turn them on and keep that way for some time depending on its location. An example of different behaviors, after sensing presence in corridors, a request is sent to the luminaires around the sensor to turn on for a short time, and

therefore predict the movement of the person with other presence sensors to follow his path. In offices, laboratories and workshops where presence is more extended and a different behavior of the luminaires is required, the light should never turn off while users are still in. In our approach this all software driven by automation rules.

Laboratories, workshops or even warehouses that have a high potential of accidents need a group of sensors that can detect and timely signal those accidents. Sensors like gas, flame, temperature, noise should monitor in real time the division and alarm the gateway in case of any anomaly in the sensors levels.

3.3 Device Management

The implementation for this work imposed the existence of a simple and lightweight object model to be used in the platform's constrained devices. For this reason, we selected LWM2M from OMA.



(a) **(b)**

Fig. 2. (a) Lightweight M2M Overall Architecture with Protocol Stack. (b) IPSO Object Representation.

The OMA Lightweight M2M architecture is composed by a LWM2M Client (M2M device/Gateway) and a LWM2M Server (M2M service/platform/application) using CoAP as the communication protocol. LWM2M provides light and compact interfaces along with an efficient data model, which together enables device management and service enablement for M2M devices. Figure 2a illustrates the Lightweight M2M architecture.

LWM2M defines the interactions between a client and a server. In order to manage the LWM2M client it's needed a object model. Regarding this subject, the IPSO Alliance created an object model based on the Lightweight M2M specification from the Open Mobile Alliance dedicated to work on constrained devices. This object

model is a solution for creating interoperability between connected devices and objects. The structure of this model is represented in Figure 2b. The Figures also show the simple addressing strategy of IPSO (object/instance/resource).

The LWM2M object specification states that an object can be a type of device and his resources are the properties that the device has.

The number of objects that IPSO defined didn't include some scenarios present in the Smart Cloud of Things Platform. Due to this, we extended the platform with several new objects using the LWM2M format and IPSO addressing strategy. The objects created and used for this work include the objects that are found in a smart building, specifically smart lightning associated objects (Luminaire object e.g.), and now any other object can be created in order to enable orchestration of a new device.

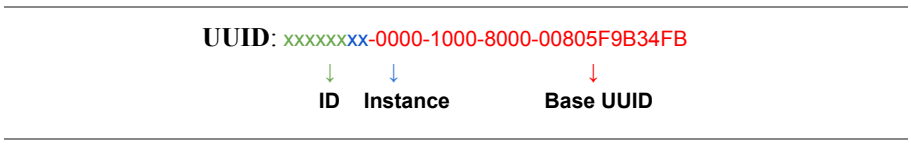
The proposed device management solution addresses a scenario where the Gateway Domain will hold the LWM2M Client logic and the Data Domain the LWM2M Server logic, due to the existence of legacy devices in the platform. The Gateway will connect several sensors and actuators, mapping them into objects in the LWM2M logic, thus enabling interaction with COTS devices.

The Device Management Component in the Data Domain will have the LWM2M Server that will be responsible for client registration and to perform management operations on the clients.

The client side of solution can be divided in two parts, MQTT/HTTP solution and OM2M solution. The MQTT/HTTP solution consists in having a LWM2M Client in the Gateway Domain, that is responsible for creating the objects and registering himself in the server. This client will also receive management actions from the server and receive real time information from the sensors/devices and update the objects automatically.

3.4 Gateways

The key point of the gateway, is to transport data between devices and the IoT platform, in our case using a MQTT connection. For that, we developed an agent that maps the services/characteristics from BLE into the objects/resources representation specified by the device management platform. To achieve this, considering that BLE uses UUIDs to identify each service/characteristic, the ID and the instance of both the objects and resources must be included in the UUID, which is done as follows:



Furthermore, in order to ensure the trustworthy of the devices and that only them are able to connect to the platform, both the gateway and devices can implement a challenge–response authentication, using a special characteristic for that purpose. This

both enables the gateway to discover new devices automatically, and to only process authorized devices.

3.5 Automation Rules

Fig. 3. CEP flow diagram example.

For making complex processing of large streams of events, WSO2 CEP was the platform chosen for this implementation. It uses Siddhi as its CEP system, Apache Storm for creating an high performance distributed platform, and provides support for several technologies and formats for data transport, including MQTT, E-mail and SMS. The processing flow for each rule is divided in 4 main elements as shown in figure 3. The event streams, which are the definition of the format and the fields of the events, are used as tables by the execution plans where queries can be made using Siddhi QL, which is similar to SQL. The receivers and publishers are responsible for, respectively, receive events and inserting them into event streams and publish events from event streams. All these elements can be controlled in a web interface provided by WSO2, but also using a SOAP interface, which is the one used by the platform described in the next section.

3.6 Building Management

The building management platform, comprises two main applications as shown in figure 4. The structure manager application, is the part of the system that holds a virtual representation of the building, in order to allow more complex selection of data in the rules. For instance, a rule can be applied to a set of specific sensor types in all the bathrooms of a specific floor in a specific building.

It is managed through an intuitive Web interface, where the manager can easily create, edit or delete buildings, floors, room types and rooms. Moreover, in each room, the manager can dynamically create areas, to which he can associate devices, with a drag-and-drop interface.

Appended to the structure manager application, there is the device manager module which is responsible for discovering and configuring devices. Internally, for each device, based on the virtual representation, it generates the topics necessary for enabling the device to publish and receive events, making sure it subscribes not only its individual topic but also the topics of the structure elements it belongs to.

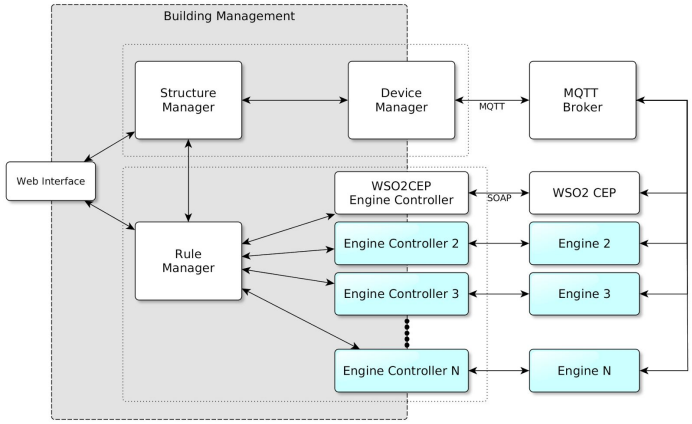


Fig. 4. Building Management Platform

Regarding to the rule manager application, although WSO2 CEP provides a lot of possibilities for creating complex queries with its powerful language, Siddhi, they are difficult to map to a Graphical User Interface. With this in mind, one of the first principles in consideration when starting this implementation, was to design a solution that would be highly extensible, with pluggable new modules. In order to achieve that, the database was implemented in a way that new modules could be added just by dropping them on a specific directory of the application. These modules just need to inherit one of the defined entities and enrich them with extended features.

All these modules can be used in a user-friendly interface for creating complex rules, for being then applied by the system on the CEP engine. However, instead of directly creating the Siddhi code for being sent to the WS02 CEP, the system first represents the whole rule in a JSON object. The reason that lead to this was, again, to allow an high extensibility of the system. By creating JSON rules in a specific format, the system is not obligated to use WSO2 CEP specifically, and thus other CEP engines can be used and even coexist. To achieve this, an engine controller must be implemented for supporting a different CEP engine, as shown in figure 4. The primary task of an engine controller is to parse the JSON rule and convert it to code supported by its engine.

4 Experimental Validation

Since this implementation is being applied in a real-world scenario, the evaluation focused in the proper functioning of a real-world prototype, considering the development of both software and hardware. The underlying hardware was composed by a quad-core server with 4 gigabytes of RAM hosting the WSO2 CEP, a Raspberry Pi 3 with the gateway agent and, finally devices consisting of two luminaires and multiple PIR sensors.

The developed prototype uses a LED luminaire powered by a RCOB-1050 LED driver by Recom, it is a constant current LED driver that outputs 1050 mA with 0-10V dimming signal input, enabling dynamic control of the brightness level..

The Nucleo-L476RG development board by STM, based on the ARM® Cortex®-M4 32-bit RISC paired up with the X-NUCLEO-IDB04A1 Bluetooth 4.0 (BLE) provided integration of the multiple devices. For this purpose we developed several devices: two of them controlling one luminaire each, one controlling a passive infrared sensor (PIR) for detecting presence, and one module with several environmental sensors. The environmental sensors report information periodically, and are able to measure: ambient light level, a temperature, humidity, contaminant gases, accustic noise and atmospheric pressure.

Each luminaire allows configuration of the Dimming type (Analog or PWM), Maximum and minimum light level, Dimming up and dimming down slope time, Logarithmic dimming resolution (softer or sharper light level transitions), and Failsafe behavior. Having a failsafe behavior prevents the interoperability of the luminaire in case of a communication breach. In other case it allows the luminaire to react to a PIR activation without the need for the remaining components.

With the aim of testing the performance of the whole system, we measured the response time between a motion detection and the corresponding action on the luminaire. Table 1 shows the results from 124 measurements in the prototype in addition to the processing times provided by the CEP system regarding the time taken to process events.

Response time	Minimum	Maximum	Average	Std. deviation
CEP Delay	5 ms	83 ms	44 ms	-
Round-trip time	68 ms	380 ms	149 ms	56 ms

Table. 1. System response time in real-world prototype.

Using an oscilloscope, we have estimated the delay in the BLE connection to be aproximatelly of 79 ms, obtained by measuring the impulse delay between the Nucleo and the gateway.

The values obtained were also compared against the simulation environment (see Table 2), which operates exactly in the same manner but is composed only of software agents (no hardware sensors or actuators). Therefore, the latency imposed by the physical communication channel will not be present. These results are important because they represent a system with a much higher number of devices (671 devices), and because they are able to characterize the performance of the software components, and in particular the CEP system and IoT platform.

Response time	Minimum	Maximum	Average	Std. deviation
Round-trip time	5 ms	233 ms	54 ms	39.32

Table. 2. System response time without physical devices.

The results prove the system's high performance, showing that even considering the latency from both the network and the Bluetooth connection, it is able to respond with a sub-second latency. The simulation scenarios also demonstrate the scalability of the system, up to 600 actuation and sensing devices.

6 Conclusions

We demonstrated a solution for easy creation of complex and fast automation in, among others, the lighting infrastructure of a building. Through an interactive web portal, a user can create complex rules that are dynamically added to a CEP system.

With the implementation and evaluation described in this paper, we have shown that the system is able to respond with a sub-second latency and thus the user have no perception of any delay. The current implementation provides the basis for creating a full featured system capable of automating efficiently and intelligently the several infrastructures of a building. For demonstration purposes it was only endowed with basic modules for creating rules for lighting control. Thus, as future work, more modules shall be added to the system for allowing the creation of more complex rules.

Additionally, manual control of devices by the user with its smartphone is also a very interesting feature, and is easily integrable in the system by requesting the building management application for temporarily disabling the automation for a specific device. Moreover, lightweight CEP engines must be implemented at the gateways in order to allow local automation and thus better failure handling.

References

1. Siemens, *Communication in building automation,* 2014
2. J. Olsson, "6lowpan demystified", Texas Instruments, Tech. Rep., 2014.
3. Wi-Fi Alliance. (2016). Wi-Fi HaLow : Low power, long range Wi-Fi, [Online]. Available: http://www.wi-fi.org/discover-wi-fi/wi-fi-halow (visited on 05/23/2016).
4. D. C. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems.* 2001.
5. LEDs Magazine,"*Straight talk sheds light on the LEDs vs linear fluorescent debate*", Aug 15, 2012
6. Alliance, Open Mobile, "Lightweight Machine to Machine (Tech Spec) Architecture V1.0" pp. 1-112, 2012
7. Eclipse, "OM2M." [Online]. Available: http://www.eclipse.org/om2m/
8. IPSO Alliance, "IPSO SmartObject Guideline," pp. 1–39, 2014.
9. B. A. G. Hillen, I. Pas, E. F. Matthijssen, and F. T. H. D. Hartog, "Remote Management of Mobile Devices with Broadband Forum's TR-069."
10. Mário Antunes, João Paulo Barraca*, Diogo Gomes, Paulo Oliveira and Rui L. Aguiar, "Smart Cloud of Things: An Evolved IoT Platform for Telco Providers", Journal of Ambientcom, Vol. 1, 1–24. doi: 10.13052/AMBIENTCOM2246-3410.111